Universitatea POLITEHNICA din București
Facultatea de Automatică și Calculatoare
Departamentul Calculatoare

Lucrare de Diplomă

Punct de acces mobil aerian pentru retele de senzori folosind drone

Autor

Ioan Deaconu

Coordonator: As. Drd. Ing. Andrei Voinescu

București, Septembrie 2014

University POLITEHNICA of Bucharest
Faculty of Automatic Control and Computers
Computer Science and Engineering Department

Diploma Thesis

Aerial Mobile Gateway for Wireless Sensor Networks utilizing drones

Author

Ioan Deaconu

Supervisor: As. Drd. Ing. Andrei Voinescu

Bucharest, September 2014

Contents

Abstract

This thesis proposes a new way of implementing a mobile gateway for Wireless Sensor Networks that simplifies applications running in remote locations, where maintenance is difficult. Wireless Sensor Networks islands need a gateway connection in order to reach the outside world, but this is difficult to provide in all instances. The solution to this problem is to use a gateway mounted on a UAV that can reach those islands and extract data from them. This has been proven to be feasible but adoption is low because of the high cost and the technical background needed to operate it. We propose a simpler, easier to operate and cheaper solution for this problem.

Keywords Wireless Sensor Networks, drones, sparrow, mobile, gateway, aerial

Acknowledgements

Foremost, I would like to express my gratitude to my advisor As. Drd. Ing. Andrei Voinescu for the continuous support that was necessarily to complete my thesis.

Besides my advisor, I would also like to give special thanks to Dan Dragomir for helping me correct this work.

My sincere thanks also goes to Adriana Drăghici, Alexandru Corneliu Olteanu and Dan Ștefan Tudose for their insightful comments and moral support.

I thank my fellow labmates in Robolab robotics group: Tudor Vișan, Andrei Mușat and Andrei Vasiliu for the sleepless nights we were working together before deadlines, and for all the fun we have had in the last four years.

Last but not the least, I would like to thank my family who have shown understanding and have given me the moral support needed.

Chapter 1

Introduction

Wireless Sensor Networks consist of small devices that can communicate with each other. They allow us to sense phenomena in the environment and act upon them. They can be used to monitor crops, to detect possible forest fires, to detect the presence of animals or vehicles in certain areas, to track and monitor doctors in hospitals etc. [6], [5]

One goal of a Wireless Sensor Network is to collect data and send it to a device called gateway. The gateway platform can be a PC connected to one of the nodes, a mobile phone or any device that can connect to a node and serve as a base-station.

Some applications are hindered by the difficulty in obtaining data from the nodes due to long distance between WSN area and base-station. Cable connection is not a possibility either but the area is accessible to an Unmanned Aerial Vehicle. The way the UAV is controlled in order to reach that area can be done by either using a remote control or using an autopilot that follows a list of predetermined waypoints specified by the user. As soon as the UAV reaches the nodes it can start retrieving and saving the data so it can be sent back home. The data collected by the drone should be accessible at any time, if the UAV is powered.

In the last ten years, integration of wireless sensor networks with unmanned aerial vehicles had been tested and proven to be successful. However, previous implementations, described in chapter 2, are complicated, difficult to operate and too expensive for the general public.

The solution that we propose is based on a very popular and easy to use drone, the Parrot AR.Drone 2.0, and the Sparrow family [15] of sensor nodes, developed at University POLITEHNICA of Bucharest.

Chapter 2

Related Work

Previous research into applications of WSNs in remote locations have proposed mobile gateways mounted on UAVs. The main focus of that research is on communication protocols and data collecting, with a lower empahsis on costs. Different systems are proposed for integrating an UAV with a WSNs, but their application to real life scenarios is limited by the high costs of the equipment used and the necessary knowledge to install and operate that equipment. The general directions of previous research into WSN and UAV integration are:

- Using node signals to perform course corrections for dynamic navigation [12]
- Using drones for node deployment, to create, expand or fix problems in the network [4]
- Data muling protocols

2.1 Standard WSN Protocols

The protocols used in common Wireless Sensor Network deployments are based on neighboring node discovery in order to build a network topology and find the best multi-hop route to the gateway. This approach works best in static environments, but in a dynamic environment or an environment were the distance between nodes is large or the time between two data packets is long, network convergence is slow or even impossible.

2.2 UAV experiments with Wireless Sensor Networks

In [13] a solution consisting of ground nodes with pre-assigned GPS positions is proposed. An RC plane with on board GPS module would record the current GPS position at which a signal was received from the nodes. The data sent by nodes contained their GPS coordinates. Combining the multiple GPS data, the plane would calculate the best path for muling the data from the network and perform the necessarily course corrections.

The advantage of using a plane in this experiment is the longer range and higher speed that it can offer as opposed to a quad-copter or a similar design. But the high speeds create the problem of maneuverability. The plane has a turning range of 400 meters while a quad-coter drone can turn on the spot.

2.3 Crop Monitoring

In [14] a system which uses a drone for crop monitoring at a vineyard is proposed. The system was comprised of a unmanned quad copter, an Arduino board with a GPRS module (used for long distance communication with the drone), ZigBee and Crossbow's TelosB as wireless sensing nodes. The drone was not controlled via the long-distance link, but through a Spektrum DX7SE 2.4 GHz remote control.

The authors demonstrate that a preprogrammed UAV can be used to monitor multiple crops where a standard WSN could not be deployed because of the unique constrains imposed by the environment.

The cost of the implementation is relatively high compared to our solution: the remote control alone costs 300$, the same price as the drone we propose, and a TelosB node costs 99$, almost 3 times as much as a Sparrowv3 node.

Another disadvantage of the system is that the data is not saved locally, but sent back to a base station where it is processed and saved. This can pose a problem in remote environments, were a base station cannot be deployed, as the system cannot function properly without one.

2.4 Aware platform

The Aware platform [10], proposed by Ays. Egül Tüysüz Erman, Lodewijk Van Hoesel and Paul Havinga from University of Twente, is a platform that integrates static and dynamic WSNs, UAVs and actuators into a disaster response situation and provides facilities for event detection, autonomous network repair by UAVs and quick response by integrated operational forces.

They use multiple UAVs to check the correct functioning of nodes and deploy new nodes that can replace damaged ones. The entire system still relies on a sink, to collect the data and to send it to a base station [9]. The base station will asses the situation and will act accordingly.

Chapter 3

Hardware Platform

This chapter presents the hardware platforms used by the solution proposed in this thesis.

Because we wanted to create not only a way of acquiring data from WSNs in remote locations, but also a simple and low cost solution, we selected the Parrot AR.Drone 2.0 as our work horse. The drone carries the mobile gateway which communicates with the nodes of the Sparrow sensors family. The drone provides several key features required by our solution, such as an embedded Linux system, a mobile platform with the possibility of autonomous flying and sufficient flight time.

In order to keep the price low and the footprint small, the Sparrow family uses a surface mounted antenna. The antenna has a gain of -1 dBi and it is perfect for applications were size is a constraint. When size can be overlooked, the range of the device can be extended by mounting an external antenna.

The cost of our solution depends on the configuration and size of the WSN islands. As previously mentioned the drone costs 300$, the price for one node is 35$, and the dongle is 45$. A high gain replacement antenna for the Sparrow nodes can cost 8$.

Example costs for two different types of networks:

- An area that needs a high density of nodes and covers a small surface does not require the nodes to have an external antenna. The system will be comprised of a drone, one dongle with an external antenna and 10 nodes. The total cost would amount to approximately 700$ give or take.

- An area that needs a low density of nodes, but covers a larger surface, or are placed in high vegetation or in a high interference area will require the nodes to be equipped with an external antenna. The system will be comprised of a drone, one dongle with an external antenna and 10 nodes,

also with external antennas. The total cost would amount to approximately 800$ give or take.

3.1 The Parrot AR.Drone 2.0

Figure 3.1 shows Parrot AR.Drone, a Wi-Fi radio controlled quad-copter built by the French company Parrot [11] . The original drone was released in 2010 and in 2012 it was replaced by version 2.0. Since the launch of the original AR.Drone, more the half a million units have been sold, making it one of the, if not, the most popular drone on the market [7]. The reason of its success is not entirely due to the relatively low price, but also because of its embedded Linux system and integrated USB port that can accommodate any device using that interface which is compatible with the Linux operating system. This makes the drone an incredibly versatile platform and can be very easily integrated with different systems.

Figure 3.1: *The Parrot AR.Drone 2.0*

Because of its popularity and versatility the Drone has a number of after-market modules that can be attached to it, such as the Flight Recorder GPS Module. This module has a built-in storage of 4GB for video recording purposes and a built in GPS receiver. This allows the drone to follow a predetermined path of waypoints and to return back from where it took off automatically, all within the limits of the Wi-Fi connection with the controlling device.

In order to properly accommodate the SparrowDongle, the hull had to be modified . The required external antenna of the dongle was mounted on top of

Image taken from [2]

the polyester cover and a small counterweight has been glued at the opposite side of the dongle. The counterweight acts as a stability ballast that keeps the drone leveled while flying and can be seen in figure 3.2.

The Parrot AR.Drone 2.0 has the following specifications:

- 1GHz, 32 bit ARM Cortex A8 processor at 800MHz
- video DSP TMS320DMC64x
- Linux 2.6.32 kernel
- 1GB DDR2 RAM at 200MHz
- USB 2.0 high speed for extensions
- Wi-Fi b,g,n
- 3 axis gyroscope with 2000°/second precision
- 3 axis accelerometer with ±50mg precision
- 3 axis magnetometer with ±6° precision
- Pressure sensor with ±10 Pa precision
- Ultrasound sensors for ground altitude measurement
- 60 fps vertical QVGA camera for ground speed measurement
- 30 fps 720p front mounted camera

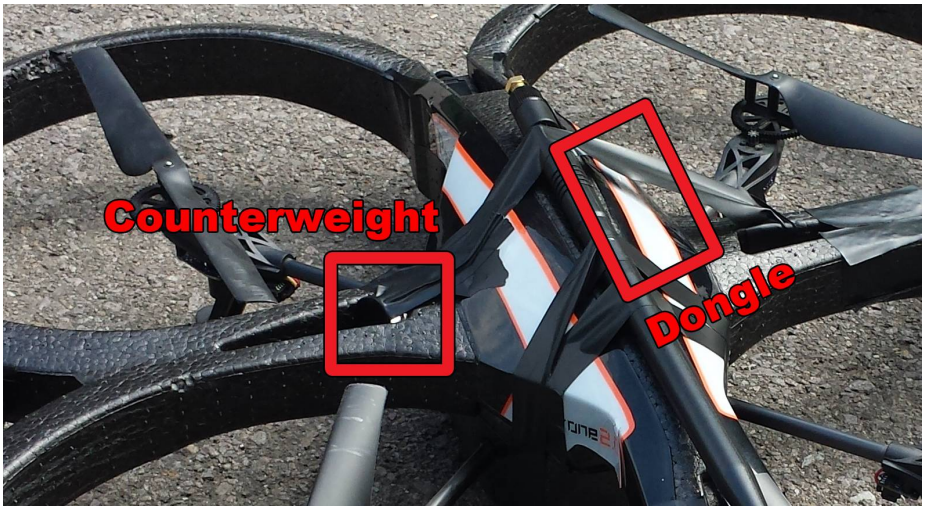

Figure 3.2: *The counterweight needed to balance the drone*

3.2 The Sparrow family

The Sparrow family, composed of the SparrowDongle and SparrowV3.2 from figure 3.3, uses the 2.4 GHz band as a medium of communication.

The main component of this family is the ATMega128RFA1. It is an 8-bit micro-controller from Atmel that has an on-chip 2.4 GHz wireless transceiver. On-chip transceivers occupy no extra PCB space and require little extra electronics to operate, making the footprint of the resulting boards very small. The on-chip transceiver allows more energy-efficient operating modes, and facilitates higher bandwidth transfers between the micro-controller's main memory and the transceiver frame-buffer, for a smaller power consumption.

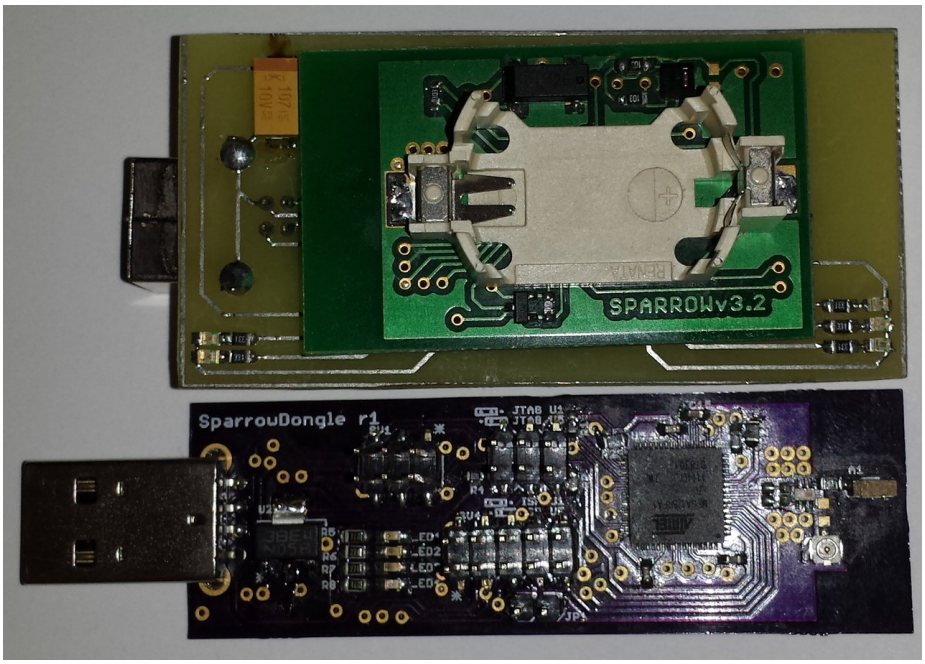

Figure 3.3: *The SparrowDongle next to the SparrowV3.2*

The signal received or sent by the wireless transceiver can be boosted by attaching an external antenna. For example, in an ideal situation, with no interferences from the outside world, an 8 dBi omni-directional antenna mounted on both communication devices would amount to around 300 meters of communication range, well over the 70 meters measured with the default antennas.

This distance is achieved with the RX and TX at full power. A higher battery life can be achieved by reducing RX and TX power, but the maximum communication range will be shortened. The reduction can be compensated by installing a high gain antenna, but this can significantly increase the cost.

3.2.1 The SparrowDongle

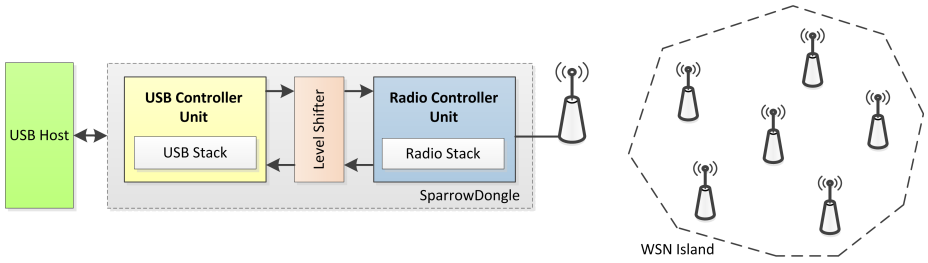

Figure 3.4: *SparrowDongle stick architecture*

The link between the wireless sensor networks and the rest of the digital world, the gateway of the Sparrow family is the SparrowDongle . The Dongle can be connected to any host that has an USB port and supports the USB CDC with ACM module (USB Communications Device Class with Abstract Control Mode).

The dongle uses an ATmega32U4 as a dedicated USB Controller unit. This design allows the RF controller to run any RF communication stack without having the USB code intrude on key timings. [15]

3.2.2 The SparrowV3.2

The other member of the Sparrow family, the SparrowV3.2 node is responsible for collecting data from the surrounding environment and sending it to the SparrowDongle. The collected data depends on the sensors attached to the SparrowV3.2. The standard implementation has light, temperature and humidity sensors. Besides this information, the nodes also collect the battery level.

The SparrowV3.2 can be modified by adding additional sensors for a better measurement of environment parameters.

The node can be powered by a non-rechargeable battery, a rechargeable one or by a super-capacitor. Even though the super-capacitor can store only a small charge, the stored energy is sufficient enough to keep the node up and running for at least a day.

The advantages of the super-capacitor over the rechargeable battery are as follows:

- it can charge and discharge almost instantaneously
- it has a very high number of charge/discharge cycles
- it does not suffer from the same aging symptoms as a battery

Image taken from [15]

- it is far less pollutant than a standard battery
- it will allow a longer maintenance-free time than a battery

The disadvantages of the super-capacitor are :

- it is bigger than battery
- it can store a much smaller amount of energy than similar size batteries
- it is very expensive
- it operates at low voltages and may require a charge pump to rise the voltage

The rechargeable battery and/or the super-capacitor can be recharged from a solar panel, a wind turbine, or any other available source of energy.

Chapter 4

Software Implementation

Our solution is divided into different modules that run independently but communicate with each other to achieve the main goal. The separation of software modules allows future features to be added easily.

There are multiple applications, running on the SparrowDongle and SparrowV3.2, running on the Parrot AR.Drone and a modified instance of the FreeFlight 2.0 application running an Android phone. The connection between them is presented in figure 4.1.

The SparrowDongle gateway is always in a listen-for-data state and dumps any data it receives on the virtual serial port emulated over USB. When it receives a data packet, it sends back an ACK message to let the SparrowV3.2 nodes know that they can begin sending the entire stored data to the it.

The SparrowV3.2 node is periodically sending a small data packet to check if a gateway is available. It stores the data accumulated during the period when no reply is received. When it receives the ACK message from a mobile gateway it starts sending the previously stored data to the gateway. The data can vary, from sensor readings to debugging information used to check the state of the Wireless Sensor Network.

The data gathered by the gateway is saved into different files in the Parrot AR.Drone's internal memory. The files also contain information such as the node identification tag and time of the transfer. The data can be accessed at any time by any device that connects to the drone's wireless network on port 4242 via FTP.

The drone also processes some of the collected data to provide real time HUD information, such as signal strength, last connection time and number of discovered nodes. This information is sent to the controlling device through a socket connection. The controlling device, PC or Android, gathers that information and displays it to the user.

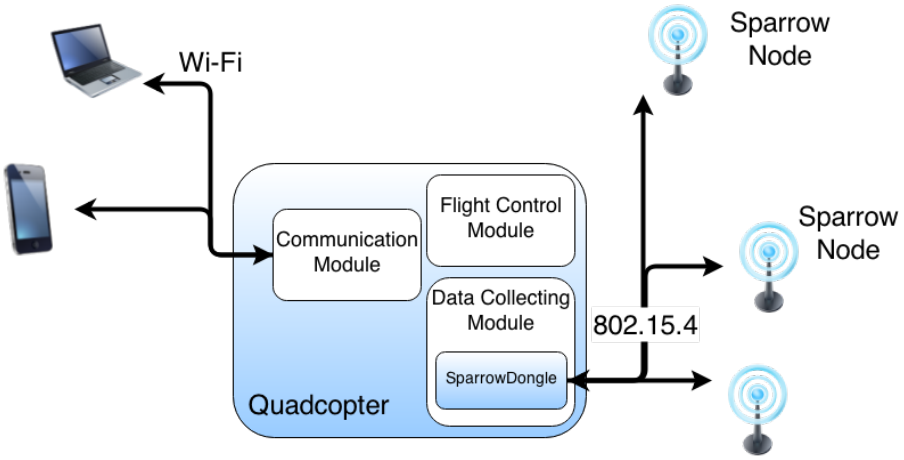

Figure 4.1: *Modules running on the drone and their connections*

4.1 Drone modules

4.1.1 The USB Module

The default software with which the drone is delivered does not implicitly recognize the dongle. This behavior was expected because the drone runs a stripped down version of Linux. In order for the drone to recognize the dongle, a module had to be compiled for the specific CPU architecture and operating system that were installed on the drone.

The required module is called cdc_acm [3]. This module is responsible for emulating serial ports over USB. The module creates a file, named /dev/ttyACM0, linked with the dongle, which can be read just like any other file.

4.1.2 The Debug Module

When performing modification to the existing code, debugging support can greatly speed up the development process. This module allows for displaying control message to the user console even when the process is running in the background. If the messages are always activated, they can slow down the execution speed of the entire application.

In order to see debugging messages, the debug option must be activated and the simple command from listing 4.1 must be run.

Listing 4.1: Simple display message command

```
p=$(pidof read) && strace -p $p
```

Enabling the debug option is just a matter of setting the *DEBUG_ON* constant to the value 1, recompiling and uploading the code to the drone. The constants needed for debugging are shown in listing 4.2.

Listing 4.2: Debug and timing defines

```
/* activates/deactivates printf debug information*/
#define DEBUG_ON 0
/* delay time in microseconds*/
#define DELAY_US 100000

/* time in milliseconds since connecting display the node*/
#define TIME_DELTA 45000
#define DEBUG_PRINT(a...) { if(DEBUG_ON) printf(a); }
```

In certain parts of the modules a delay is needed in order to wait for an action to be executed. The value *DELAY_US* can be changed to any value, but you must be careful in doing so. A small delay sends data more often but it uses a lot of processing power. A big delay could be too slow for the data to be usable. We have chosen a delay of 100 ms, for 10 data updates per second.

Also, the nodes are memorized for the period *TIME_DELTA*. During this period, the node will be saved and informations about it will be sent to the user. If after *TIME_DELTA*, no new data has been received from the node, it is deleted from the list.

4.1.3 The Data Collecting Module

The module saves the collected data into the drone's internal memory and pases the data on to the communication module, which displays on the controller interface certain data items like: the number of nodes currently connected to the dongle, the signal strength, dongle connection status etc.

This module contains some extra features that are designed to make the solution more user friendly and easier to extend in the future.

4.1.4 Modules intercommunication

The memory area in which the information sent to the user is saved is shared between this module and the communication module. The interaction method between these two modules resembles the consumer-producer problem, where the Data Collecting Module is represented by the producer and the Communication Module is represented by the consumer.

A important issue with this approach is synchronization and the avoidance of data races. These are prevented with the use of a mutex construct that only allows one thread at a time to access the data. Example is depicted in listing 4.3.

Listing 4.3: Data Collection use of mutex

```
pthread_mutex_lock(&data_lock);
add_node_data(get_current_timestamp(),read_data + 7);
pthread_mutex_unlock(&data_lock);
```

Similarly, the mutex is used in the Communication Module when it consumes data.

4.1.5 Fault tolerance

Because the Dongle is connected to an USB port on a machine that has a lot of vibrations, it might disconnect / reconnect for a very short period of time. This module has been designed to cope with multiple USB disconnects and reconnects without the need to reset the drone. This information is vital, because you can check if the dongle is still connected to the drone without the need to inspect it visually or to connect to a debug terminal.

Besides the possible USB dongle disconnects, an out of range signal may be experienced. If this happens, the drone will hover until the connection is reestablished.

4.1.6 The Communication Module

All of the information gathered by the Data Collecting Module would be useless if it cannot be accessed easily.

This module, as the name suggests, handles the communication of this crucial information back to the user.

Being a different module, with different attributions than the Data Collecting Module, it has an entire Linux process dedicated to it for 3 important reasons:

1. The approach of having a process per module allows the modules to run independently of each other.

2. The Data Collecting Module can collect the data from the Dongle as soon as this is available.

3. If the Communication Module stops working, the Data Collecting Module can keep collecting data, so complete failure of the system is avoided.

4. System processes can be restarted in case of failure.

4.1.7 Socket with connection reset

The communication is done through socket connections listening on port 8888. The server running on the drone accepts only one connection at a time.

If a connected client decides to disconnect before or while a write operation is in progress, a $SIGPIPE$ error signal will be thrown, stopping all the modules. This is prevented by ignoring the signal, forcing the write action to return a $EPIPE$, and exiting gracefully.

The main process will use the callback accept_socket_connection to reestablish a new connection. Once a connection is established, it will send information once every $DELAY\_US$ microseconds. The program was configured and tested with a 100ms wait period that translates to ten updates per second.

This delay is required because if there was no delay the communication would occupy too much processor time both on the drone and on the controlling device.

4.1.8 JSON Encoding of Data

JSON [1] is an open-standard that uses text to encode data. It is an alternative to XML. Derived from the JavaScript scripting language, it is a language-independent data format available in most programing languages.

JSON is best suitable for this application as a data encode format because it is data oriented, unlike XML which is document oriented. Also, it is very easy to encode because it has a code like structure, the result being smaller than the XML alternative. Another advantage is that all devices can decode it.

The informations encoded by the drone are:

- Dongle connection status
- An array containing node data
  - Node unique ID
  - Last connection time of the node to dongle
  - The power of the received signal

4.2 SparrowV3.2 module

Due to the communication protocol implemented, the SparrowV3.2 node uses very little power when not connected to the drone. The solution is similar to the one described by Cardei et al[8]. The Sparrow node sends a small packet at a fixed interval.If the packet is received by the dongle, it will send back a specific ACK just to the node that sent the packet. When the node receives the ACK, it will try to send all the stored data to the drone, starting from the oldest entry to the newest one. The data is stored in a circular linked list of a fixed size. If the list is full, the oldest entity is always replaced by new data.

The drone saves the received data, locally in files that have the name comprised of the timestamp of the current session and the unique id of the node.

4.3 Android application modules

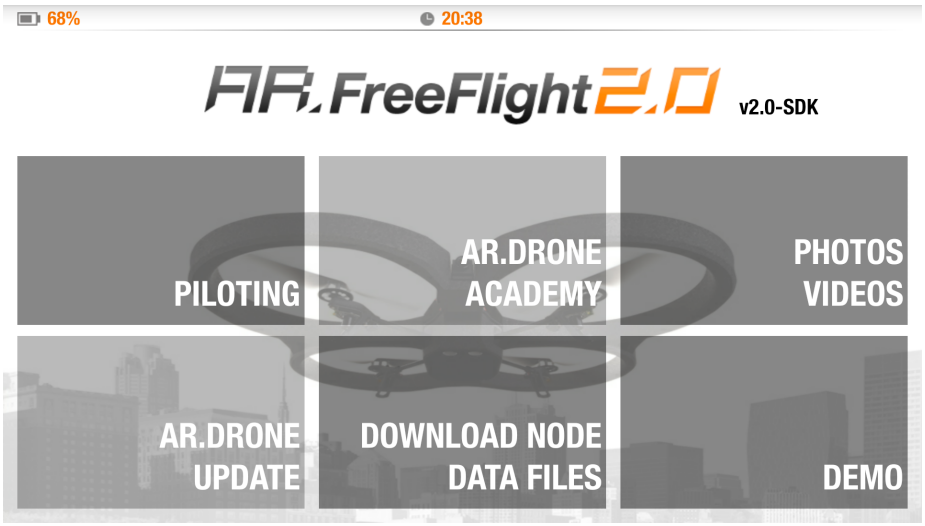

Figure 4.2: *AR Freeflight modified application*

Being an open-source platform we modified the AR Freeflight 2.0 Android application to communicate with the new modules installed on the drone and added a new feature on the main screen, seen in figure 4.2.

Android imposes the use of the background process class AsyncTask when using communication protocols such as http, ftp or sockets. This prevents the UI process from remaining stuck in communication and not responding to user actions.

The class offers 5 very important methods that can be overwritten, 4 running on the main UI process, that prepare data before and after execution, publish progress or simply cancel at any step, and 1 running on the background process.

4.3.1 Display information module

The Piloting screen of the application has been modified to display the received data from the drone.

The information displayed is comprised of the state of the dongle, the number of connected nodes and for each node the unique id, last connection time and the signal strength. Up to 9 nodes sorted in descending order after their signal strength are displayed. Figure 4.3 is an example of how the informations are displayed.

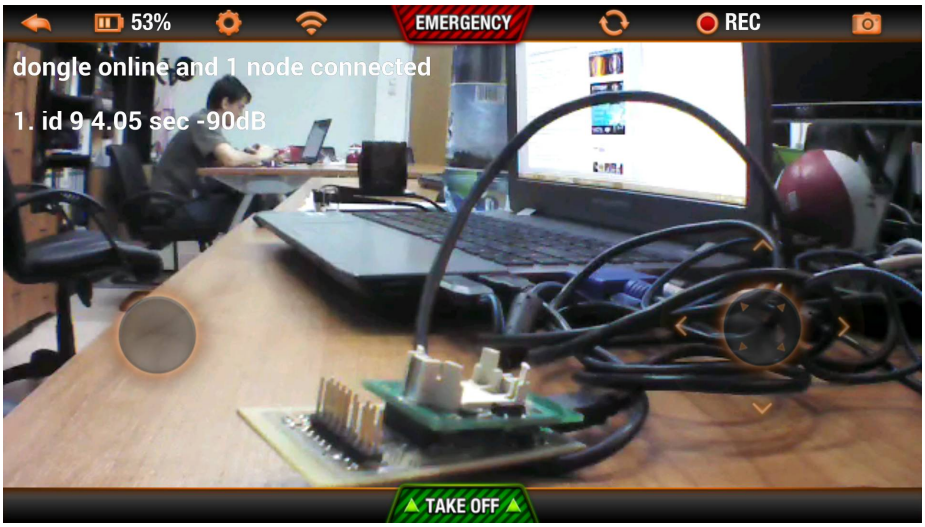

Figure 4.3: *AR Freeflight modified Piloting Screen*

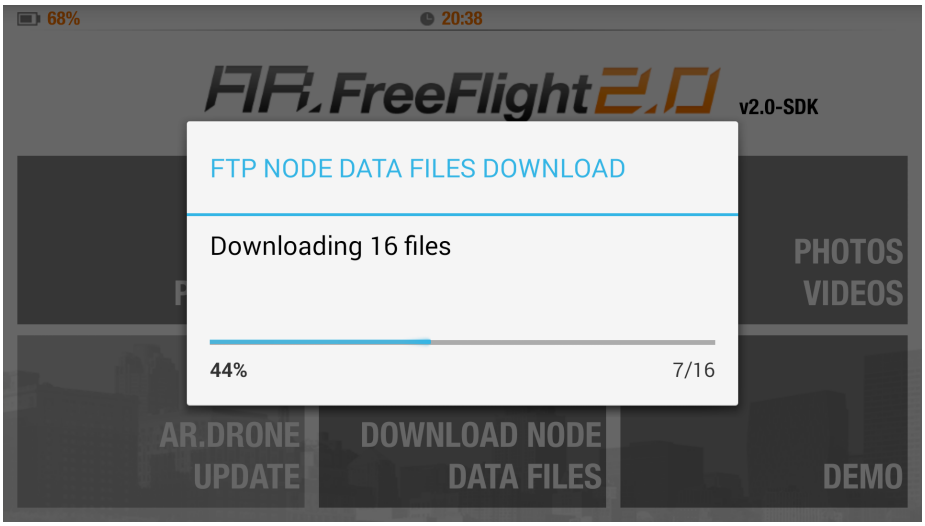

Figure 4.4: *AR Freeflight FTP downloading files*

4.3.2 FTP communication module

The drone has a built-in FTP server that can be configured to allow access to any folders/files on the drone. We have configured the drone so that the folder in which the data is saved can be accessed at any time using port 4242 by any device that has FTP client capabilities.

This feature was added to the Android application as well, to have a better out of the box experience.

The application will download all the files from the drone to the folder *ftp_node_data*

from the local storage of the user's Android device and display the progress as in figure 4.4.

Chapter 5

Evaluation

The tests that we have conducted highlight the strengths of the platform but also reveal some of its weaknesses.

The characteristics that we considered to be most important are the maneuverability of the drone with the added components, the maximum range of the dongle and the number of packets sent by the nodes and not received.

5.1 Scenario description

The test scenario is relatively simple. Using the drone, we try to connect and collect the data from different nodes while measuring the identified characteristics.

We conducted multiple tests using a dongle with an 8 dBi antenna, two SparrowV3.2 nodes, one with a standard antenna and one with a 8 dBi antenna, were placed on a hill with the antenna directed upwards for a maximum range test. Another three nodes with 3 dBi external antennas were placed inside of a building. The nodes were configured to send a packet every second. The controlling device of the drone was a Samsung Galaxy S4.

Because both the drone and nodes use the 2.4 GHz band, the drone was configured to use channel 11 and the nodes were configured to use channel 1 in order to prevent any signal interference.

The packet loss test was performed using 4 scenarios. In all scenarios a node with an 8 dBi antenna and a node with -1 dBi antenna was used.

- Scenario 1: The drone hovered at a distance between 0 and 50 meters from the nodes
- Scenario 2: The drone hovered at a distance between 50 and 100 meters from the nodes

- Scenario 3: The drone hovered at a distance between 100 and 150 meters from the nodes
- Scenario 4: The drone hovered at a distance between 150 and 200 meters from the nodes

5.2 Results

5.2.1 Packet loss

The results of the test based on the 4 scenarios can be seen in figure 5.1. The high gain antenna performs excellent, for up to 100 meters distance there is no packet loss and at almost 200 meters distance, the packet loss is just 14 %. On the other hand, even though the -1 dBi could send packets at almost 200 meters, the loss is 75 %. The acceptable distance up until a -1 dBi antenna should be used is for a 100 meters connection because of a much smaller packet loss of 24%.

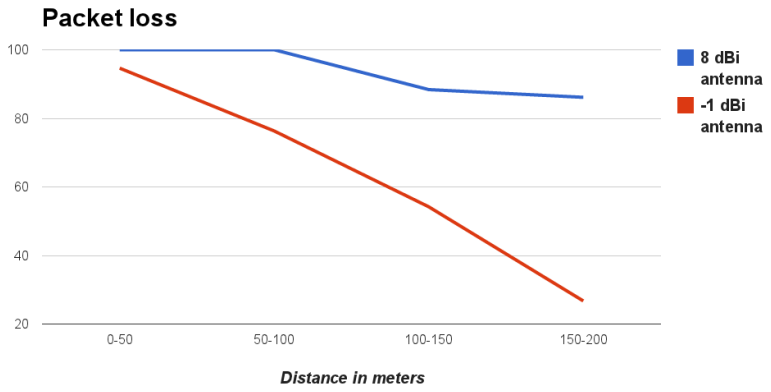

Figure 5.1: *The results for the packet loss test*

5.2.2 Signal range

The drone was able to find the nodes but, as depicted in figure 5.3, the obstructed nodes had a much smaller communication range than the ones placed on the hill. The range test results proved that the drone could receive a signal at a maximum distance of 325 meters from the node with the 8 dBi external antenna located on the hill and a clear line of sight, highlighted in figure 5.2. The other node on hill did manage to send data up to 100 meters. In the case of the obstructed nodes, even though they had a 3 dBi antenna, the distance did not exceeded the 65 meters mark. After this distance, the number of lost packets was to big to be

Figure 5.2: *Measured signal distance*

able to properly communicate with the nodes, even though, occasionally, at 90 meters, some packets were received.

The top mounted antenna worked, but for a better signal quality, the antenna should always be positioned on the bottom of the drone. In this way, the antenna will have a clear path to send and receive signals and not be blocked by the drone's body and its avionics. The problem with this particular drone model is that on its bottom there are sensors that help it determine the ground speed and ground altitude. If we place the antenna on the bottom it would obstruct some of the sensors.The other possibility, a vertical antenna, will increase wind resistance and make the drone too unstable.

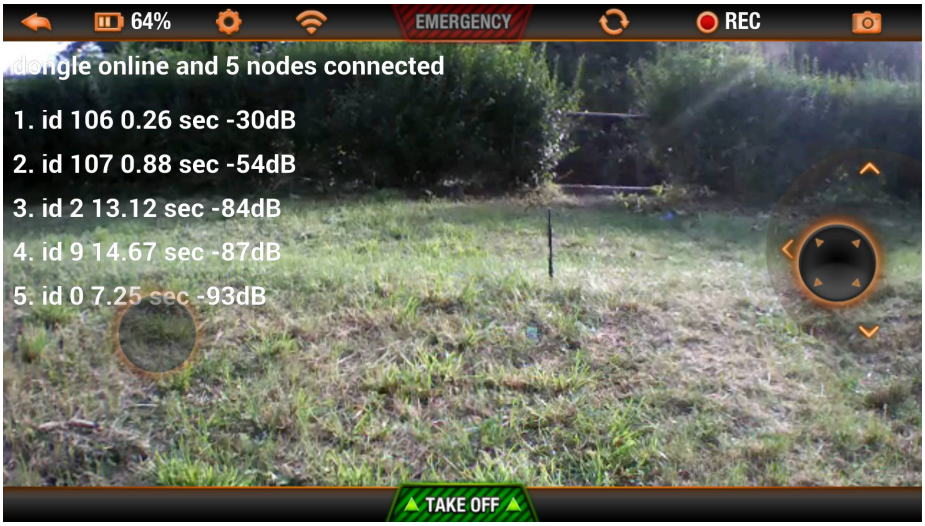

Figure 5.3: *Parrot discovering new nodes before takeoff*

5.2.3 Drone stability

Even though the antenna was mounted on top of the body, because it is a high gain antenna the signal received was very strong.

The dongle is mounted on the side of the drone due to the position of the USB connector. Because the dongle is not centered on the drone, a counterweight had to be glued on the opposite side on the outer shell to maintain the balance of the drone. The antenna, shown in figure 5.4, extended the signal range but also added weight.

The drone was relatively stable during tests, but a better stability and flight control could be obtained if the dongle were to be made smaller and a lighter antenna were to be used.

Figure 5.4: *Top mounted antenna for better signal quality*

5.2.4 Maximum height and maneuverability

The total added weight is 75 grams. Even though it does not sound that much, it does have a substantial effect on the drone. The maximum height it can reach is 50 meters versus the 75 meters it can reach without the added weight. The maneuverability is also affected, the drone response not being as sharp as before, but it is still good.

5.2.5 Problems

The kernel module needed for the dongle does not recognize the dongle if it is plugged in the drone when its powering up. The fix is to power up the drone and

then plug in the dongle, but this is more difficult than it might appear because every time this action is performed the hull must be repositioned.

Chapter 6

Conclusions

Wireless Sensor Networks are expanding more and more because they help make our lives easier by giving us information about our surroundings. However, the standard way of creating wireless sensor network islands is not always feasible.

The main goal of this thesis was to bring an alternative solution to the problem of how data can be obtained from Wireless Sensor Network islands. We have proven that a slightly modified low-cost drone offers a good range, being able to communicate with the nodes from a big distance. The communication range is greatly affected by the antenna gain and the objects that obstruct the line of sight from the drone to the node.

The solution, a truly mobile gateway is a viable one and can be implemented with a relatively low cost. The technical knowledge necessary to operate the system is not more complicated than knowing how to use a smartphone.

6.1 Feature Work

The proposed system can be improved in a number of ways. A feature that can be used in conventional wireless sensor networks is to determine the source of a communication failure. If the gateway detects that the network has a communication problem and not all of the previous nodes can be reached, a drone can use this information to search and find exactly which nodes are working properly and which nodes are not.

The Parrot AR.Drone 2.0 can perform autonomous flight with a GPS module, but only while it is still in the range of the Wi-Fi connection. This module can allow the drone to fly without the need to still be connected through Wi-Fi to a controlling device. A different autonomous flight mode can be implemented without a GPS module if the signal strength of the nodes is used to perform flight correction and determine the speed and direction of the drone.

Bibliography

[1] Json Standard. http://www.json.org/xml.html. [cited at p. 16]

[2] Parrot Drone. http://img.clubic.com/04862120-photo-parrot-ar-drone-2-0.jpg. [cited at p. 7]

[3] USB ACM. http://wiki.openmoko.org/wiki/CDC_ACM, March 2010. [cited at p. 13]

[4] Ian F Akyildiz, Weilian Su, Yogesh Sankarasubramaniam, and Erdal Cayirci. Wireless sensor networks: a survey. *Computer networks*, 38(4):393–422, 2002. [cited at p. 4]

[5] Th Arampatzis, John Lygeros, and S Manesis. A survey of applications of wireless sensors and wireless sensor networks. In *Intelligent Control, 2005. Proceedings of the 2005 IEEE International Symposium on, Mediterrean Conference on Control and Automation*, pages 719–724. IEEE, 2005. [cited at p. 3]

[6] Aline Baggio. Wireless sensor networks in precision agriculture. In *ACM Workshop on Real-World Wireless Sensor Networks (REALWSN 2005), Stockholm, Sweden*, 2005. [cited at p. 3]

[7] ALEX BRACETTI. The 10 Best Drones You Can Buy Right Now. http://www.complex.com/pop-culture/2013/03/10-cool-drones-you-can-buy-right-now/parrot-ardrone-20, March 2013. [cited at p. 7]

[8] Mihaela Cardei and Ding-Zhu Du. Improving wireless sensor network lifetime through power aware organization. *Wireless Networks*, 11(3):333–340, 2005. [cited at p. 16]

[9] Aysegül Tüysüz Erman, LV Hoesel, Paul Havinga, and Jian Wu. Enabling mobility in heterogeneous wireless sensor networks cooperating with uavs for mission-critical management. *Wireless Communications, IEEE*, 15(6):38–46, 2008. [cited at p. 5]

[10] Anibal Ollero, Markus Bernard, Marco La Civita, Lodewijk van Hoesel, Pedro J Marron, Jason Lepley, and Eduardo de Andres. Aware: Platform for autonomous self-deploying and operation of wireless sensor-actuator networks cooperating with unmanned aerial vehicles. In *Safety, Security and Rescue Robotics, 2007. SSRR 2007. IEEE International Workshop on*, pages 1–6. IEEE, 2007. [cited at p. 5]

[11] AR Parrot. Drone. *Available: ardrone. parrot. com*, 75, 2012. [cited at p. 7]

[12] Chris Savarese, Jan M Rabaey, and Jan Beutel. Location in distributed ad-hoc wireless sensor networks. In *Acoustics, Speech, and Signal Processing, 2001. Proceedings.(ICASSP'01). 2001 IEEE International Conference on*, volume 4, pages 2037–2040. IEEE, 2001. [cited at p. 4]

[13] Steven K Teh, Luis Mejias, Peter Corke, and Wen Hu. Experiments in integrating autonomous uninhabited aerial vehicles (uavs) and wireless sensor networks. 2008. [cited at p. 5]

[14] João Valente, David Sanz, Antonio Barrientos, Jaime del Cerro, Ángela Ribeiro, and Claudio Rossi. An air-ground wireless sensor network for crop monitoring. *Sensors*, 11(6):6088–6108, 2011. [cited at p. 5]

[15] Andrei Voinescu, Dan Tudose, and Dan Dragomir. A lightweight, versatile gateway platform for wireless sensor networks. In *Networking in Education and Research, 2013 RoEduNet International Conference 12th Edition*, pages 1–4. IEEE, 2013. [cited at p. 3, 10]

List of Figures

Listings